Design Technology for Systems-on-Chip

Raul Camposano and Don MacMillen
Synopsys, Inc

Abstract: Advancing technology impacts design along several vectors. Interconnect delay became dominant in many designs at 0.18μm; to obtain timing closure, it was necessary to unify synthesis and placement. Moving forward, increasing degradation of signal integrity will be caused by capacitive cross coupling, by inductive effects and by several other physical effects. This will require the integration of fast and accurate analysis that can drive avoidance and correction of signal integrity problems primarily in routing, as well as in synthesis and placement. Advancing technology also means increasing complexity. Verification is particularly affected by technology as exemplified by the ever-increasing simulation needs. Using hundreds of millions of devices effectively will be possible only by reusing pre-designed intellectual property (IP) effectively and by addressing system-level issues in EDA. This presentation poses EDA solutions to these challenges, gives concrete examples, and argues that complete solutions, rather than point tools, will increasingly and justifiably dominate the EDA field.

1. INTRODUCTION

Electronic Design Automation (EDA) is one of the key enablers of the semiconductor industry [1]. No chip is designed without EDA. Conversely, semiconductors drive EDA technology. Throughout the last four decades this has happened mainly along two vectors: technology and complexity.

Technology drives EDA in many ways. Early on, EDA efforts concentrated on capturing and editing artwork. This has led to automatic placement and routing. Eventually synthesis raised the design level to the Register-Transfer Level (RTL). Synthesis, placement and routing are enabled by multiple technology constraints: a logic library, usually in the form of standard cells of the same height and similar size which simplifies

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35597-9_40
M. Robert et al. (eds.), *SOC Design Methodologies*

placement and routing, decouples technology from logic and enables synthesis. Furthermore, until recently interconnect was assumed to have negligible delay and digital circuits were modeled without "analog" effects such as cross talk. Power consumption issues were limited to the power grid. All that has changed. Chips today typically are designed using large pre-designed blocks referred to as Intellectual Property (IP) as well as standard cells which may be up to six orders of magnitude smaller [2].

Interconnect can no longer be ignored, since for many designs under 0.25μm interconnect delay actually becomes larger than circuit delay [3]. Capacitive coupling may produce severe cross talk. Power issues are not limited to power grid design: optimizing overall power consumption may be the foremost goal of the design, electro migration and hot spots need to be prevented, and leakage may become a serious power contributor second only to dynamic power. Section 2 examines how technology is driving EDA today. We will show how design technology is radically changing to cope with the changes in semiconductor technology.

The second vector, which has driven EDA relentlessly, is complexity. Moore's Law epitomizes complexity growth: device density on a chip doubles every 18 months. Design tools need to keep pace, making capacity increases a perennial requirement. Nowhere has this reality been more pronounced than in simulation, particularly in logic simulation. The simulator has to keep up not only with the increasing size of the designs to simulate but also with an exploding numbers of vectors to simulate. Moore's Law has yielded a hundred times the number of devices per chip in the last 12 years, but the number of vectors needed to simulate a chip to achieve functionality confidence has increased ten thousand times during that same period. A simulator thus has to cope with a complexity that has grown by six orders of magnitude in slightly over a decade. The argument for the development of formal verification techniques, emulators and smart test benches is easy to understand in this context [4]. Section 3 shines more light on how verification techniques are being driven to cope with complexity.

But there is another profound change affecting EDA besides technology. As chips become more complex, they increasingly encompass larger and larger subsystems or complete electronic systems. The electronic design of these "Systems-on-Chip" (SoC) requires system knowledge. Conversely, designing systems becomes increasingly designing chips. Given this increasingly strong correlation, how does system design automation (SDA) relate to EDA? System design has been automated much less than electronic design. Furthermore, SDA tends to be much more domain specific than EDA. Finally, system design today is mostly IP-based. Processors enable the implementation of large parts of functionality in SW. The concept of IP is further extended by defining "platforms," which are basic architectures

geared towards specific families of applications. In section 4 we examine the influence of system design on EDA, expanding the arguments made above.

The paper concludes summarizing the main trends we see in design technology for systems-on-chip.

2. TECHNOLOGY DRIVES EDA

A state-of-the-art design flow consists of some 50 individual design tools. These can be divided into design and verification tools. Technology drives changes in both. In this section we will focus on how technology is influencing design tools and drives EDA towards integrated design flows by migrating "up" in the design tool chain.

Figure 1 shows a simplified design flow for cell-based digital Application Specific Integrated Circuits (ASICs). This kind of methodology was widely used for digital ASICs down to 0.25μm processes. It assumes that you can effectively separate different phases of the design such as logic design, placement, routing, etc., which in turn relies on the fact that physical effects can be "abstracted" (ignored) at higher levels. For example, wire delays can be assumed to be zero or can be modeled statistically during logic design.

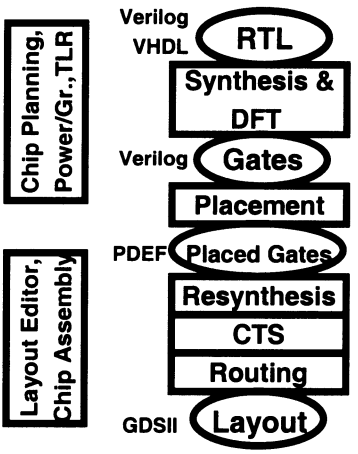

Figure 1. Simplified Design Flow

As technology progressed below 0.25mm these assumptions do not longer hold. Wire delay becomes more important than circuit delay in many cases. Figure 2 shows the delay of a 43μm long wire in aggressive copper technology compared with the smallest gate delay.

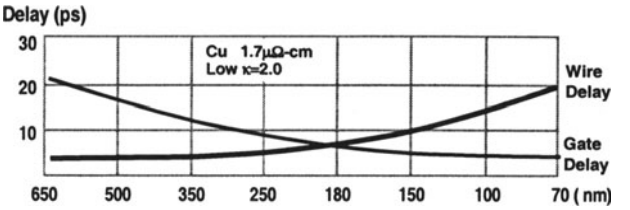

Figure 2. Wire Delay vs. Gate Delay

As a consequence, wire delays can't be ignored in logic design any more, meaning that synthesis needs to have a good estimate for wire delays. The initial solution we adopted long ago was the so-called "wire load model," which is a statistical model of delay dependent on the design size. Clearly this doesn't work well below 0.25μm. Our solution was to do placement and synthesis together so that wire lengths can be accurately estimated during combined synthesis and placement.

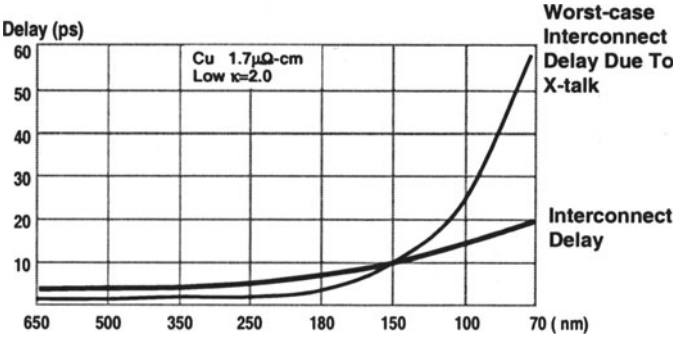

Figure 3. Interconnect Delay due to Crosstalk

Moving forward, as the technology node approaches 0.10μm, other physical effects come into play. Capacitive coupling gives rise to cross talk. Cross talk can be measured as additional delay before a signal stabilizes at the driven value. Worst-case cross talk for a pair of 43μm long wires is illustrated in Figure 3 and compared to interconnect delay in the absence of cross talk.

Cross talk can no longer be ignored in logic design for technology nodes below 0.18μm. Computing cross talk requires knowledge of the exact position of the wires involved. Avoiding cross-talk can be achieved by several mechanisms: signals can be offset so that they switch at different times, wires can be placed further apart, driver strengths can be increased, additional buffers can be inserted, etc. Hence, we expect to see a much tighter integration of synthesis, placement and routing.

In addition to the two examples given above, there are several additional physical effects which will require changes in the design flow. We will only mention some of the most important ones here. Inductivity, typically ignored on chips so far, becomes noticeable for long wires at high speed such as busses of several mm of length operating at more than 1-2GHz [5]. This is already a problem for microprocessor design. The design flow will have to incorporate delays due to inductive effects.

Power dissipation of CMOS circuits so far has been well approximated by exclusively modeling the dynamic power. As we approach 0.10μm, leakage power will become significant. Again, EDA needs to incorporate models for leakage and then minimize it by, for example, using multiple threshold-voltage libraries [6].

Another example involves masks. The physical layout, which defines the patterns for the fabrication process is not used directly for mask manufacturing. To compensate for limited optical resolution, optical proximity correction adds patterns to the masks. Additional resolution enhancements can be achieved using phase-shift masks. In EDA, this resolution enhancement technology is being incorporated into integrated layout verification / mask production tool suites [7].

We hope we made our point: technology drives EDA. As we move towards smaller technology nodes, more physical effects need to be taken into account during design. In turn, this drives the integration of design tools, ultimately towards an integrated RTL to layout flow and a layout to mask flow.

3. DEALING WITH COMPLEXITY

Nowhere is the effect of complexity felt more than in design verification. Moore's Law has enabled a growth of one order of magnitude (10x) every 6 years in device density and the same growth in design size. The complexity of "system behavior" (the number of input vectors necessary to cover the function of a system) has been increasing by roughly two orders of magnitude over the same period. Functional verification is proportional to the product of the design size and system behavior complexity. For this section, we selected functional verification to show how EDA is dealing with complexity.

Functional verification is based mainly on simulation. How has simulation speed kept up with the thousand-fold increase of simulation need every 6 years (Figure 4)? For a simulator running on a single general-purpose computer, simulation speed increases stem from mainly two sources:

Faster simulators. In our experience, Verilog simulators have provided a speed gain of approximately two times per year over the last six years, resulting in a total speedup of 50 times. This has been achieved through migration from gate to RTL simulation and through optimizations in the simulators algorithms.

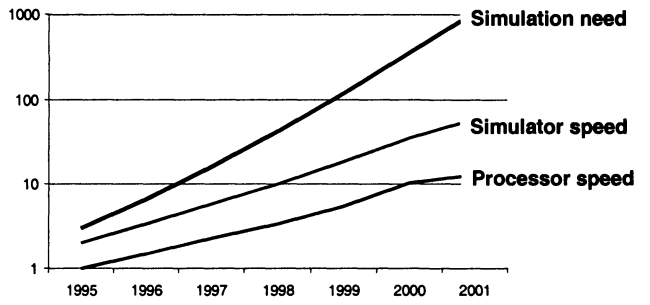

Figure 4. Normalized Simulation Speed 1995=1

Increasing processor speed. Processor speed is a complex function of many variables such as clock frequency, memory speed, architecture, etc. Over the last 6 years clock speeds have increased by a factor of 10-12. Memory speed has not kept up, while architectural changes have delivered more performance. Overall, processor speed has increased by one order of magnitude in the last six years.

Thus, functional simulation speed has increased by roughly 10*50=500 times over the last 6 years, approximately half the rate of the thousand-fold need. Further speedup can be achieved by hardware acceleration or emulation. Emulation can be 4 to 5 orders of magnitude faster than functional simulation. Emulation speed has scaled with the chip (mostly FPGA based) speed increases dictated by Moore's law. Another way of achieving speedup is parallelism. Computer "farms" running thousands of independent simulation runs in parallel provide a cost-effective way of obtaining a 3 order of magnitude simulation speedup.

It needs to be emphasized that simulator speedup has come at the cost of losing the detail that gate-level simulation provides. For example, it is not possible to do an accurate timing simulation at the RTL level. As a consequence, static timing analysis, which scales essentially with the size of a design only, has become adopted as a standard to determine and sign-off timing in SoC design.

Another interesting trend is based on the observation that generating and feeding billions of simulation vectors to a simulator can be as or more time consuming than the simulation itself. Test bench automation aims at

optimizing this process and at effectively monitoring the simulation of a test bench.

A simulation cannot be exhaustive (with the exception of very simple cases). This raises the question of how much simulation is enough or how good is a test bench. To measure this, the concept of simulation coverage is introduced and coverage tools monitor how much of a design has been "covered" by a simulation (a test bench).

But simulation alone, since it can't be exhaustive for large designs, is not enough. Formal techniques have emerged as a powerful aid to functional verification. Equivalence checking asserts whether two net lists (or RTL descriptions) implement the same Boolean function. Although this is an NP-complete problem, in practice equivalence checking works on many large designs up to millions of gates. Equivalence checking allows comparing designs against a "golden" model, which has been extensively simulated and is assumed to be functionally correct. Thus it relieves the designer from the need of having to simulate after changes have been made to a design.

Property checking asserts that a given property holds for a given implementation under all conditions. For example, a protocol may require for a design to output an acknowledge signal before a given number of cycles after receiving a request signal. If this can be verified formally, there is no need to simulate for this property. There is another interesting link between simulation and property checking. Property checking can prove that a set of given states can never be reached. Hence, when measuring the state coverage of a given test bench, those states can be excluded, something simulation alone wouldn't allow.

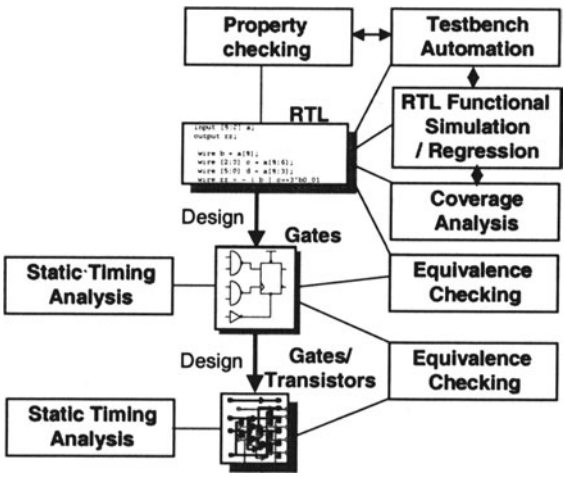

Figure 5. Functional Verification Flow

Figure 5 shows a verification flow using the tools discussed above. Again, we hope we made our point: EDA is dealing with complexity by constantly increasing tool capacity (such as simulator speed), by innovation (e.g., making formal verification practical), and by integrating tools into flows which leverage the combined strength of all tools (such as decreasing the need for simulation by formal verification and measuring coverage).

4. SYSTEM LEVEL DESIGN NEEDS TO BE AUTOMATED

With the advent of SoCs, the electronic system design and chip design become more tightly integrated. The system designer needs to know what can be done on a chip to deliver good designs and the chip designer needs to understand the systems he/she is designing. Increased communication among system and chip designers can be achieved in many ways, two of which are especially relevant to EDA: system level design (SLD) tools and the use of pre-designed blocks or sub-systems called IP.

SLD tools enable a system designer to enter, verify and possibly implement a system in a way that he/she understands. Several possibilities are shown in Figure 6. Chip designers can use the same SLD tools as a "formal" specification of the design. The SLD tool may be used to execute (simulate) the design and possibly to automatically generate a first implementation, which can then be optimized. EDA technology has addressed multiple levels of abstraction. At the system level however, abstractions tend to be domain specific. For example, data-flow models are used in digital signal processing, hierarchical final state machines model reactive control, and protocols can be described in various languages. EDA has addressed system-level issues in some of these domains, but the SLD market is still embryonic. Recent developments, such as the widespread interest in SystemC and the increasing number of companies offering SLD technology, show that this technology is maturing and is beginning to appeal to a broader audience. System level design automation is a logical next step for EDA.

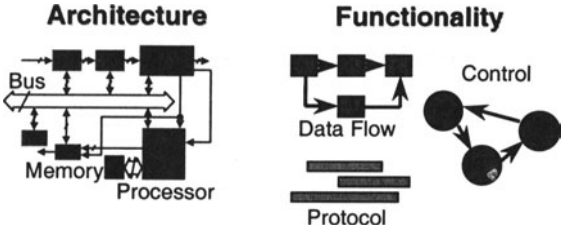

Figure 6. System level Design Models

There are no SoC designs today that don't (re)use IP. Processors, memories, standard interfaces, buses, etc. have become ubiquitous in SoCs. Processors in particular enable the implementation of system functionality in software. System design becomes increasingly influenced by the processor or combination of processors used as a "platform."[8]_For example, a wireless handset may require a processor and a DSP for application processing and a second DSP and a micro controller for communication processing. System design then consists of integrating some hardware blocks into such a platform and implementing most functionality in software.

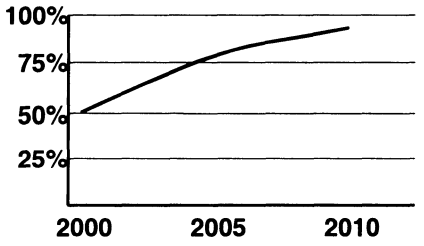

Figure 7. IP Reuse as a % of a SoC

The reuse of IP is also the most effective way of addressing complexity in design (Figure 7). IP is increasingly becoming one of the main differentiators for system and chip design. EDA and SLD need to use the same IP and platforms to enable effective communication between the two levels.

Our final point is that SLD needs to be automated: The introduction of SLD tools and the increasing (re)use of IP are clear signs that this is happening.

5. CONCLUSIONS

EDA continues to change to meet the needs of electronic design. In this paper, we showed how the main forces affecting this change are technology, complexity, and the advent of Systems-on-Chip. As a consequence, (1) more technology effects are being taken into account earlier in the design process, (2) individual design tools are being integrated into design and verification flows, (3) tool capacity is keeping pace with complexity, (4) innovation results in smarter tools, (5) system level design is gaining increased attention, and (6) IP reuse and design platforms are becoming pervasive. We expect these trends to continue as long as integrated circuit manufacturing keeps moving to smaller technology nodes.

6. REFERENCES

[1] Don MacMillen, Mike Butts, Raul Camposano, Dwight Hill, and Thomas Williams, "An Industrial View of Electronic Design Automation", *Transactions on CAD, Volume 19, No.21*, IEEE, Dec.2000, pp.1428-1448

[2] Warren Savage, John Chilton, Raul Camposano, "IP Reuse in the System on a Chip Era", *Proceedings of the 13th ISSS*, Madrid, Spain, September 20-22, 2000, pp.2-7

[3] Jason Cong, "An Interconnect-Centric Design Flow for Nanometer Technologies", *Proceedings of the IEEE*, VOL. 89, No. 4, April 2001, pp. 505-528.

[4] Bob Bentley, "Validating the Intel Pentium 4 Microprocessor", *Proceedings of the 38th Design Automation Conference*, Las Vegas, June 18-22, 2001, pp. 244-248

[5] Alina Deutsch, Paul Coteus, Gerard Kopcsay, Howard Smith, Christopher Surovic, Byron Krauter, Daniel Edelstein, Phillip Restle, "On-Chip Wiring Design Challenges for Gigahertz Operation", *Proceedings of the IEEE*, VOL. 89, No. 4, April 2001, pp. 529-555

[6] Luca Benini and Giovanni De Micheli, "Dynamic Power Management, Design Techniques for CAD Tools", Kluwer Academic Publishers, Boston, 1998.

[7] Andrew Kahng and Y.C. Pati, "Subwavelength Lithography and it Potential Impact on Design and EDA", *Proceedings of the 36th Design Automation Conference*, New Orleans, June 21-25, 1999, pp. 799-804.

[8] K. Keutzer, S. Malik, A.R. Newton, J.M. Rabaey, and A. Sangiovanni-Vincentelli, "System-Level Design: Orthogonalization of Concerns and Platform-Based Design", *Transactions on CAD, Volume 19, No.21*, IEEE, Dec.2000, pp.1523-1543